📝

PROPOSTA

🗄️ Sistema IoT de Gestão de Estoque de Uniformes com Balança (Edge–Cloud)

Júlia Calixto Rosa - NUSP: 13749490
Rafael - NUSP: 15510017

---

1. Descrição do Projeto

1.1 Problema Inicial

Empresas que distribuem uniformes enfrentam a falta de controle e visibilidade sobre o estoque, o que causa:

- Custos ocultos por excesso ou falta de itens;
- Desvios e perdas sem rastreamento;
- Dificuldade em auditar retiradas;
- Processos manuais ineficientes (planilhas, contagens físicas).

1.2 Abordagem Proposta (Foco na Balança)

A solução consiste em uma **Estação IoT Inteligente com Balança** que registra automaticamente as variações de **peso** dos uniformes para inferir a quantidade em estoque.

Como os uniformes têm **pesos pré-estabelecidos** por tamanho/modelo, a variação de massa detectada na balança é usada para calcular as unidades adicionadas (entrada) ou retiradas (saída) e atualizar o estoque em tempo real, assim como reportar as alterações.

Os dados são processados localmente em um dispositivo **Edge** (Raspberry Pi) e enviados à **Nuvem** (AWS) para análise e visualização em um dashboard.

Isso permite ao gestor financeiro:

- **Visibilidade em Tempo Real** da quantidade de uniformes em estoque;
- **Controle de Consumo** e reposição otimizada, com alertas por limite;
- **Redução de Perdas** por descontrole;
- **Eliminação de Contagens Manuais**.

---

2. Arquitetura Fim a Fim

2.1 Visão Geral

O sistema segue a arquitetura IoT **Edge–Cloud**, cobrindo o ciclo completo:
Balança/Sensores → ESP32 → Edge (Raspberry Pi) → Nuvem (AWS: IoT Core, Lambda, DynamoDB) → InfluxDB → Dashboard (Grafana)

Fluxo de Dados Revisado:

1. **Ação na Estação:** O funcionário adiciona ou retira uniformes da balança.
2. **ESP32/Balança:** O módulo amplificador da balança converte o sinal analógico do peso em digital e o **ESP32** (Controlador) recebe essa variação de peso (massa) e o envia para o

processamento no Raspberry Pi.

3. **Edge (Raspberry Pi):** Recebe o dado de peso, **calcula a variação (Δ_{peso})** e, com base nos pesos unitários pré-configurados para cada tipo/tamanho de uniforme, **infere a quantidade de unidades** adicionadas ou retiradas. O Raspberry Pi envia a informação da **variação de unidades** para o AWS IoT Core.
4. **Nuvem (AWS IoT Core, Lambda, DynamoDB, SNS):**
   - **AWS IoT Core:** Recebe os dados de variação de unidades.
   - **Lambda:** Centraliza **a lógica de controle**. Recebe a variação de unidades, atualiza a **quantidade total em estoque** no DynamoDB, e determina se a variação foi uma **Entrada** (peso aumentou) ou **Saída** (peso diminuiu).
   - **DynamoDB:** Armazena o registro de transações e a **quantidade atualizada do estoque**.
   - **Lambda (Feedback):** Envia um sinal de retorno para o ESP32 para acionar os LEDs/Buzzer. Paralelamente, verifica os **limites** de estoque (mínimo/máximo) e envia um **Alerta SNS** se for necessário (ex: estoque baixo ou alto).
5. **Feedback Local (ESP32):** Recebe o sinal da Nuvem (Lambda) e aciona:
   - **LED Verde:** Confirma a **Entrada** (uniforme adicionado).
   - **LED Vermelho:** Confirma a **Saída** (uniforme retirado).
   - **LED Azul:** Status **Aguardando**
   - **Buzzer:** Confirmação sonora da transação.
6. **Visualização (InfluxDB → Grafana):**
   - **InfluxDB:** Puxa os dados de quantidade e transação para séries temporais.
   - **Grafana:** Cria a visualização (Dashboard) do **estoque em quantidade**, **limites** (inferior/superior) e **alertas**.

---

3. Materiais e Componentes

Componentes	Função (Revisada)
4 Células de Carga + amplificador HX711	**Sensor principal da Balança**. Medição da massa dos uniformes.
ESP32	Controlador da Balança, responsável por ler o dados, acionar LEDs/Buzzer e comunicar com o Raspberry Pi.
Raspberry Pi	Computador principal (**Edge**). Processa a variação de peso para unidades e envia dados processados para a AWS IoT Core.
Display LCD 16x2	Exibição de instruções e *status*
LEDs (Vermelho, Verde, Azul) + Buzzer + Cabos	Feedback visual e sonoro de **Saída**, **Entrada** e **Aguardando**.
Resistor + Transistor + Protoboard + Fonte de Alimentação 5V	Circuitos auxiliares e alimentação.
Gabinete / Estrutura da Balança	Suporte físico para a célula de carga e componentes.

Cronograma (sujeito a alterações)

FASE 1: Hardware e Edge (Semana 1 - Até 19/10/2025)

Foco na montagem física da balança, leitura de peso estável com o ESP32 e comunicação com o Edge (Raspberry Pi).

Tarefa	Detalhes
1.1. Aquisição/Organização de Materiais	Garantir todos os componentes: Célula de Carga, HX711, ESP32, Raspberry Pi, LEDs, Buzzer, etc.
1.2. Montagem da Balança e Calibração	Montagem da célula de carga e módulo HX711. Desenvolvimento de um *sketch* inicial no ESP32 para ler o peso e **calibrar** o sistema.
1.3. Lógica do *Edge* (Raspberry Pi - **Inferência**)	Desenvolvimento do script no RPi para: a) Receber a variação de peso do ESP32 (via serial ou MQTT local). b) Aplicar a lógica: Δpeso → Δunidades (ex: 500g→1 uniforme).
1.4. Implementação do *Feedback* Local	Integrar LEDs e Buzzer ao ESP32. Implementar a lógica local para exibir **LED AZUL (Aguardando)**.
1.5. Teste de Conectividade do *Edge*	Configurar a conectividade Wi-Fi no RPi e simular a publicação de dados de variação de estoque para um **tópico de teste MQTT** (ainda sem AWS).

FASE 2: Nuvem (AWS IoT Core e Lambda) (Semana 2 - Até 26/10/2025)

Tarefa	Detalhes
2.1. Configuração do AWS IoT Core	Criar o **Thing**, baixar certificados (`.crt`, `.key`) e criar a **IoT Policy** de mínimo privilégio.
2.2. Configuração do DynamoDB	Criar a tabela `estoque_uniformes_iot` com **Partition Key** (`device_id [S]`) e **Sort Key** (`timestamp [N]`).
2.3. Criação e Teste da AWS IoT Rule	Criar a *IoT Rule* que aciona a Lambda a partir do tópico de telemetria do RPi (ex: `balanca/estoque/telemetria`).
2.4. Desenvolvimento da Lógica Lambda (Cálculo de Estoque)	Código do Lambda Completa: a) Receber: `device_id`, `ts`, e `delta_unidades`. b) **Buscar** o estoque atual no DynamoDB. c) **Calcular** o novo estoque (`Estoque Atual` + `delta_unidades`). d) **Atualizar** o estoque no DynamoDB.
2.5. Integração Lambda - Feedback - SNS	Configurar a Lambda para: a) Enviar um *publish* de retorno para o ESP32 (tópico de controle) para acionar o LED Verde/Vermelho/Buzzer. b) Implementar o **Alerta SNS** se o estoque atingir limites (ex: `if novo_estoque < Limite_Minimo: alertar`).

FASE 3: Visualização (InfluxDB e Grafana) (Semana 3 - Até 02/11/2025)

Tarefa	Detalhes
3.1. Configuração do InfluxDB Cloud	Criar organização, *Bucket* (`iot_estoque_uniformes`) e gerar o **API Token** (Read/Write), conforme o guia do termostato.
3.2. Integração Lambda - InfluxDB	Ajustar as variáveis de ambiente da Lambda para incluir os dados do InfluxDB e verificar se a função `send_to_influxdb` está enviando a `quantidade_estoque` como um *field*.
3.3. Configuração do Grafana	Configurar a **Data Source** no Grafana (Cloud ou Local) usando a URL, Organização e Token do InfluxDB, com *Query Language* **Flux**.
3.4. Criação do Dashboard de Estoque	Desenvolver os painéis principais: a) **Gráfico de Séries Temporais** do Estoque

	(Quantidade). b) **Single Stat** para o *Estoque Atual*. c) **Alertas** (visualização de limites inferior/superior).
3.5. Revisão e Teste de Carga	Testar o fluxo completo simulando várias retiradas e entradas para garantir que o Grafana, InfluxDB e DynamoDB estão em sincronia.

FASE 4: Finalização e Documentação (Semana 4 - Até 09/11/2025)

Tarefa	Detalhes
4.1. Estabilização e Tratamento de Erros	Implementar *try/excepts* e *logging* robustos no RPi, ESP32 e Lambda para tratamento de falhas de conexão ou erros de leitura da balança.
4.2. Documentação do Código e Arquitetura	Revisar os códigos do ESP32, RPi e Lambda (incluindo variáveis de ambiente e tópicos MQTT) para clareza e comentários. Finalizar o diagrama de arquitetura.
4.3. Testes de Validação Final	Teste de Aceitação do Usuário (simulação de uso real) para garantir que a variação de peso está gerando a contagem de unidades correta.
4.4. Apresentação e Entrega do Projeto	Preparar a demonstração e o relatório final.